Very efficient learning of structured classes of subsequential functions from positive data

Adam Jardine, Jane Chandlee, Rémi Eyraud, Jeffrey Heinz

JMLR: Workshop and Conference Proceedings 34:94–108, 2014 Proceedings of the 12th ICGI

Very efficient learning of structured classes of subsequential functions from positive data

Adam Jardine AJARDINE@UDEL.EDU
Dept. of Linguistics and Cognitive Science, University of Delaware, Newark, USA

Jane Chandlee CHANDLEE@ASEL.UDEL.EDU
Center for Pediatric Auditory and Speech Sciences, Nemours Biomedical Research, Wilmington, DE, USA

Rémi Eyraud REMI.EYRAUD@LIF.UNIV-MRS.FR
QARMA team, Laboratoire d'Informatique Fondamentale, Marseille, France

Jeffrey Heinz HEINZ@UDEL.EDU
Dept. of Linguistics and Cognitive Science, University of Delaware, Newark, USA

Editors: Alexander Clark, Makoto Kanazawa and Ryo Yoshinaka

Abstract

In this paper, we present a new algorithm that can identify in polynomial time and data using positive examples any class of subsequential functions that share a particular finite-state structure. While this structure is given to the learner *a priori*, it allows for the exact learning of partial functions, and both the time and data complexity of the algorithm are linear. We demonstrate the algorithm on examples from natural language phonology and morphology in which the needed structure has been argued to be plausibly known in advance. A procedure for making any subsequential transducer onward without changing its structure is also presented.

Keywords: transduction, onwardness, identification with polynomial bounds

1. Introduction

One of the most important grammatical inference results for learning string transductions is OSTIA [Oncina et al., 1993], which identifies in the limit total subsequential functions. Its behavior is not unlike that of RPNI [Oncina and García, 1992], upon which it is based. de la Higuera [2010] provides an up-to-date treatment of OSTIA. More recent work on learning string transductions has focused on learning the entire class of regular relations or even larger classes [Clark, 2011].

This paper focuses on *proper subclasses* of subsequential functions and presents an algorithm that identifies each class in the limit from positive data. There are three main reasons why this result is interesting. First, OSTIA is unable to learn partial functions exactly, while the classes studied in the present work include partial functions. Second, the algorithm we provide is able to learn in *linear* computation time and *linear* data with respect to the size of the target transducer. This is an improvement over OSTIA, which, while polynomial, requires cubic computation time. Third, this algorithm is applicable to infinitely many classes that contain infinitely many transductions.

The reason why this algorithm is possible is because each proper subclass comes with a significant amount of prior knowledge. In particular, we define a subclass of subsequential functions in terms of a finite-state structure. Essentially, the states and the transitions of the transducer are provided, and only the output labels on the transitions are missing. In this way, this analysis is similar to the one by Heinz and Rogers [2013], where the structure of the automata is fixed in advance to define an efficiently learnable class of formal languages.

The proposal is also similar to OSTIA-D [Oncina and Varò, 1996] and OSTIA-R [Castellanos et al., 1998], which use *a priori* information about the domain (OSTIA-D) and range (OSTIA-R) of the target transduction. In particular, OSTIA-D takes as input a deterministic finite-state acceptor (DFA) and adds a merging condition to OSTIA based on the states of this DFA. (Similarly, OSTIA-R takes a DFA representation of the range as input.) Like our proposal, OSTIA-D learns subclasses of subsequential functions, including partial functions, but since it uses OSTIA as a base, the time complexity is still cubic, and no complexity analysis of its characteristic sample exists to our knowledge.

The degree of requisite prior knowledge may seem undesirable, especially for cases where it is not available. However, the domain is a regular language and if it belongs to any one of several sub-regular classes, there are algorithms that can learn it from positive data. But even if that is not possible, there exist scientific fields where such prior knowledge *is* available, and in those fields it thus makes sense to use it. We provide examples from linguistics, in particular phonology and morphology. Specifically, we show that such a known structure characterizes the Input Strictly k-Local functions [Chandlee, 2014; Chandlee et al., to appear]. This class, which Chandlee shows includes significantly many phonological and morphological processes, represents one way to generalize the Strictly k-Local formal languages [McNaughton and Papert, 1971] (which are identifiable in the limit from positive data [García et al., 1990]) to transductions. Phonological processes which are not Strictly k-Local involve long-distance dependencies, and we show that these too are learnable, again if the learner is provided *a priori* knowledge of the relevant finite-state structure.

This paper is organized as follows. §2 presents the mathematical notations that will be used throughout. §3 introduces subsequential finite-state transducers (SFSTs) in two forms: the traditional definition and a minor variant of this definition (*delimited* SFSTs) that we use. This section also introduces onward transducers, which will be crucial for the learning algorithm. §4 shows how to convert any delimited SFST into an onward one without changing its structure. §5 presents the learning paradigm of identification in polynomial time and data [de la Higuera, 1997]. §6 introduces output-empty transducers, which represent the learner's *a priori* knowledge of the structure of the target transducer. §7 presents the Structured Onward Subsequential Function Inference Algorithm (SOSFIA). §8 proves that it identifies the target subclasses in the sense of de la Higuera [1997]. §9 presents several demonstrations of the SOSFIA using linguistically-motivated examples. §10 concludes.

2. Preliminaries

Let Σ denote a finite alphabet of symbols. Σ^n and Σ^* denote the sets of strings of symbols from Σ of length n and of any length, respectively. Let λ denote the empty string and $|w|$

the length of string w. If u and v are strings, uv denotes their concatenation, and $u^{-1}w = v$ iff $w = uv$. The *prefixes* of w are $\texttt{pref}(w) = \{u \in \Sigma^* \mid (\exists v)[uv = w]\}$.

A *stringset* (sometimes called a *language*) is any subset of Σ^*. The size of a finite stringset S is the sum of the lengths of its elements and is denoted $|S|$. We extend the notion of prefixes to stringsets with $\texttt{pref}(S) = \cup_{w \in S} \texttt{pref}(w)$. The *shared prefixes* of S are the prefixes shared by all strings in S: $\texttt{sh\_pref}(S) = \cap_{w \in S} \texttt{pref}(w)$. The *longest common prefix* ($\texttt{lcp}$) of S is then $\texttt{lcp}(S) = w \in \texttt{sh\_pref}(S)$ such that $\forall v \in \texttt{sh\_pref}(S) : |v| \leq |w|$. We set the longest common prefix of the empty set to be λ.

Given an input alphabet Σ and an output alphabet Δ, a *relation* from Σ to Δ is a subset of $\Sigma^* \times \Delta^*$. Given a relation t, we use $\texttt{dom}(t)$ to refer to its left projection and $\texttt{image}(t)$ its right projection. If, for all $w \in \Sigma^*$, $(w, v), (w, v') \in \tau \Rightarrow v = v'$ then τ is a *function*, and we often write $t(w) = w'$ instead of $(w, w') \in t$. A function t is *total* iff for all $w \in \Sigma^*$ there exists $v \in \Delta^*$ such that $(w, v) \in \tau$. A function that is not total is *partial*.

For any function $t : \Sigma^* \to \Delta^*$ and $w \in \Sigma^*$, we define the *tails* of w with respect to t as $\texttt{tails}_t(w) = \{(x, v) \mid t(wx) = uv \wedge u = \texttt{lcp}(t(w\Sigma^*))\}$. Two strings $w, w' \in \Sigma^*$ are *tail-equivalent* with respect to t iff $\texttt{tails}_t(w) = \texttt{tails}_t(w')$, in which case we write $w \sim_t w'$. Clearly, $\sim_t$ is an equivalence relation which partitions Σ^*. A function t is *subsequential* iff $\sim_t$ partitions Σ^* into *finitely* many blocks.

3. Representations of Subsequential Functions

3.1. Traditional Subsequential Transducers

We give here the usual definition of subsequential transducer:

Definition 1 *A* subsequential finite-state transducer (SFST) *is a 6-tuple* $\langle Q, q_0, \Sigma, \Delta, \delta, \rho \rangle$ *where* Q *is a finite set of states,* $q_0 \in Q$ *is the initial state,* Σ *and* Δ *are finite alphabets of symbols,*[1] $\delta \subseteq Q \times \Sigma \times \Delta^* \times Q$ *is the transition function,* $\rho : Q \to \Delta^*$ *is the output function, and the following holds: if* $(q, \sigma, w, r), (q, \sigma, v, s) \in \delta$ *then* $(r = s) \wedge (w = v)$*, i.e. the transition function is deterministic.*

The transition and output functions of a SFST may be partial. We extend the transition function to δ^* recursively in the usual way: $(q, \lambda, \lambda, q) \in \delta^*$; if $(q, u, v, q') \in \delta^*$ and $(q', \sigma, w, q'') \in \delta$ then $(q, u\sigma, vw, q'') \in \delta^*$.

The relation described by a SFST τ is $R(\tau) = \{(w, vv') \mid (\exists q)[(q_0, w, v, q) \in \delta^* \wedge \rho(q) = v']\}$. However, since they are deterministic, SFSTs only describe functions: for every SFST, each element of Σ^* corresponds to at most one element of Δ^*. Therefore we write $\tau(w) = v$ instead of $(w, v) \in R(\tau)$. We note that SFSTs compute *total* functions iff $\forall q_i \in Q, \forall \sigma \in \Sigma$, there exist $q_j \in Q, v \in \Delta^*$ such that $(q_i, \sigma, v, q_j) \in \delta$ and $\rho(q)$ is defined. The size of a SFST $\tau = \langle Q, q_0, \Sigma, \Delta, \delta, \rho \rangle$ is $|\tau| = |Q| + \sum_{(q,\sigma,u,q') \in \delta} |u|$.

A well-known result is that for every subsequential function t, there is a subsequential transducer τ which computes it ($R(\tau) = t$), and likewise for every subsequential transducer τ, $R(\tau)$ is a subsequential function [Oncina and García, 1991].

1. Notice that Σ can be equal to Δ, in which case we only specify one alphabet in the definition of a SFST.

3.2. Delimited Subsequential Transducers

Here we present an alternative representation of subsequential functions, which we call *delimited subsequential transducers* and which are a slight variant of traditional SFSTs.

Definition 2 *A* delimited subsequential finite-state transducer (DSFST) *is a 6-tuple* $\langle Q, q_0, q_f, \Sigma, \Delta, \delta \rangle$ *where* Q *is a finite set of states,* $q_0 \in Q$ *is the initial state,* $q_f \in Q$ *is the final state,* Σ *and* Δ *are finite alphabets of symbols,* $\delta \subseteq Q \times (\Sigma \cup \{\rtimes, \ltimes\}) \times \Delta^* \times Q$ *is the transition function (where* $\rtimes \notin \Sigma$ *is a special symbol indicating the 'start of the input' and* $\ltimes \notin \Sigma$ *is a special symbol indicating the 'end of the input'), and the following hold:*

1. *if* $(q, \sigma, u, q') \in \delta$ *then* $q \neq q_f$ *and* $q' \neq q_0$,
2. *if* $(q, \sigma, u, q_f) \in \delta$ *then* $\sigma = \ltimes$ *and* $q \neq q_0$,
3. *if* $(q_0, \sigma, u, q') \in \delta$ *then* $\sigma = \rtimes$ *and if* $(q, \rtimes, u, q') \in \delta$ *then* $q = q_0$,
4. *if* $(q, \sigma, w, r), (q, \sigma, v, s) \in \delta$ *then* $(r = s) \wedge (w = v)$.

Pictorially, the initial state of a DSFT has no incoming transitions (1) and exactly one outgoing transition whose input label is $\rtimes$ (3) which leads to a nonfinal state (2); the final state of a DSFT has no outgoing transitions (1) and every incoming transition comes from a non-initial state and has input label $\ltimes$ (2); and the DSFT is deterministic on the input (4).

Extending the transition function as before, the relation described by a DSFST τ is $R(\tau) = \{(w, v) \mid (q_0, \rtimes w \ltimes, v, q_f) \in \delta^*\}$. Again, the deterministic condition (requirement 4) guarantees that DSFSTs define functions.

The difference between DSFSTs and SFSTs is that the delimiters $\rtimes, \ltimes$ are employed to mark the beginnings and ends of input strings when computing the transducer function. Oncina et al. [1993] observe that an 'end of the input' delimiter can be used in place of the output function ρ without changing the class of functions being described. Therefore, the only real difference introduced here is the use of the 'start of the input' delimiter. It is trivially true that the class of functions describable with DSFSTs is exactly the class representable by SFSTs (i.e., the subsequential functions).

As explained in Section 4, the reason we introduce the 'start of the input' delimiter is so that we can fix a structure with a DSFST and identify a canonical representation of the function it describes without changing this structure.

We end this section with the following definitions: A state q in a DSFST τ is *reachable* iff there exists $w \in \Sigma^*, v \in \Delta^*$ such that $(q_0, \rtimes w, v, q) \in \delta^*$. A state q in a DSFST τ is *useful* iff there exists $w \in \Sigma^*, v \in \Delta^*$ such that $(q, w \ltimes, v, q_f) \in \delta^*$ and q is reachable. A transducer is *trimmed* iff every state other than q_0 and q_f is *useful*.

3.3. Onward Transducers

An important concept for learning subsequential functions is the property of *onwardness*, which we define here for DSFSTs.

Definition 3 *A DSFST* $\tau = \langle Q, q_0, q_f, \Sigma, \Delta, \delta \rangle$ *is onward iff the outgoing transitions of each noninitial state share no nonempty prefix.*

$onward(\tau) \stackrel{def}{=} \big(\forall q \in (Q - q_0)\big)\big[\texttt{lcp}\big\{w \in \Sigma^* | (\exists a \in \Sigma, r \in Q)[(q, a, w, r) \in \delta]\big\} = \lambda\big]$

Intuitively, a transducer is onward iff there is no delay in writing the output strings. As the input symbols are consumed from left to right, the output is written the moment it is determined. Onward SFSTs are discussed in detail in Oncina et al. [1993], as onwardness plays a large role in the OSTIA learning algorithm. There, as here, onwardness helps establish canonical forms for subsequential transductions.

Figures 1 and 2 (from Oncina and García [1991]) illustrate the differences between SFSTs and Onward SFSTs: these SFSTs represent the same function but Figure 1 is onward while Figure 2 is not onward, and cannot be made so without modifying its structure. For comparison, Figure 3 is a DSFST and Figure 4 is an onward DSFST, and both represent the same function shown in Figures 1 and 2.

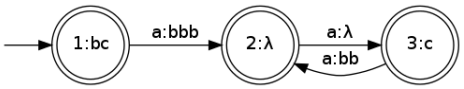

Figure 1: Onward SFST

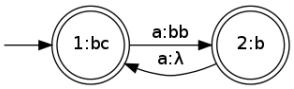

Figure 2: SFST

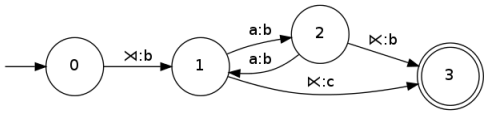

Figure 3: A DSFST

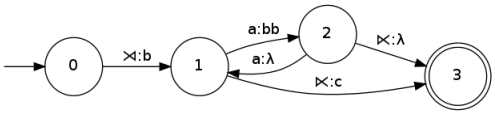

Figure 4: An onward DSFST

4. Deriving an Onward DSFST

We prove the following theorem which says for any DSFST τ, there is an onward DSFST with the exact same structure which recognizes the same function described by τ.

Theorem 4 *For any trimmed DSFST $\tau = \langle Q, q_0, q_f, \Sigma, \Delta, \delta \rangle$, there is an onward trimmed DSFST $\tau' = \langle Q, q_0, q_f, \Sigma, \Delta, \delta' \rangle$ such that $R(\tau) = R(\tau')$ and $(q, a, u, q') \in \delta$ iff there exists $u' \in \Delta^*$ and $(q, a, u', q') \in \delta'$.*

The proof of the above theorem makes use of a function `push_lcp` which takes as arguments a transducer τ and a state q of τ and returns a transducer τ' in which the longest common prefix of the outputs of the transitions leaving q is pushed as a suffix onto the outputs of the transitions entering q (if they exist).

Definition 5 (`push_lcp`) $\texttt{push\_lcp}\big(\langle Q, q_0, q_f, \Sigma, \Delta, \delta \rangle, q \in Q\big) = \langle Q, q_0, q_f, \Sigma, \Delta, \delta' \rangle$ *where*

$$\begin{aligned} \delta' = \quad & \big\{(q, \sigma, x^{-1}ux, q) \mid (q, \sigma, u, q) \in \delta\big\} \\ \cup \quad & \big\{(q, \sigma, x^{-1}u, q') \mid (q, \sigma, u, q') \in \delta\big\} \\ \cup \quad & \big\{(q', \sigma, ux, q) \mid (q', \sigma, u, q) \in \delta\big\} \\ \cup \quad & (\delta \setminus \{(q_1, \sigma, u, q_2) \mid q_1 = q \vee q_2 = q\}) \end{aligned}$$

and $x = \texttt{lcp}\Big(\big\{u \mid (\exists q', \sigma)[(q, \sigma, u, q') \in \delta]\big\}\Big)$.

In this definition, the first part of the union corresponds to reflexive transitions of the state q: the output of the transition is modified such that the `lcp` x of the outgoing transitions of q is not a prefix anymore, but is instead added as a suffix in order not to modify the transduction (see proof of lemma 6). The second part of the union concerns the non-reflexive outgoing transitions from q: the `lcp` is erased from the outputs of all of these transitions. The third part concerns the incoming transitions of q: the `lcp` is added as a suffix to the outputs of all these transitions. Finally, the last part of the union ensures that of the original transitions, only the unmodified ones are still present in the new set of transitions.

First we observe that pushing the `lcp` does not change the function described by a DSFST.

Lemma 6 *Let $\tau = \langle Q, q_0, q_f, \Sigma, \Delta, \delta \rangle$ be a trimmed DSFST. For all $q \in Q$, we have $R(\tau) = R(\texttt{push\_lcp}(\tau, q))$.*

Proof Let $q \in Q$ and $(v, w) \in R(\tau)$. Then there exist $\sigma_1, \ldots, \sigma_n$ in Σ such that $v = \sigma_1 \ldots \sigma_n$, $\forall i, 1 \leq i \leq n$ $(q_i, \sigma_i, u_i, q_{i+1}) \in \delta$, $q_1 = q_0$, $q_{n+1} = q_f$, and $w = u_1 \ldots u_n$. Consider $\tau' = \texttt{push\_lcp}(\tau, q)$.

As the result trivially holds for $q = q_0$, suppose $q \neq q_0$ and let $w' = \tau'(v)$. Since $q \neq q_0$ and τ is trimmed, there is always at least one incoming transition into q and hence a place for the `lcp` to be pushed back to. Since `push_lcp` only modifies the output string of some transitions, it is the case that $w' = u'_1 \ldots u'_n$, with $\forall i, 1 \leq i \leq n$ $(q_i, \sigma_i, u'_i, q_{i+1}) \in \delta'$. Let $x = \texttt{lcp}(\{u : (q, \sigma, u, q') \in \delta\})$ and $i_1, \ldots, i_k$ $(k < n)$ the indices such that $q_{i_j} = q$, for $1 \leq j \leq k$. We have $u'_{i_j - 1} = u_{i_j - 1} x$ and $u'_{i_j} = x^{-1} u_{i_j}$. Thus $w' = u_1 \ldots (u_{i_1 - 1} x)(x^{-1} u_{i_1}) \ldots (u_{i_k - 1} x)(x^{-1} u_{i_k}) \ldots u_n = w$ ■

Notice that this proof is not possible without the initial transition with the start delimiter. In Figure 2, applying something like `push_lcp` to state 1 modifies the represented function: for instance, (λ, bc) is replaced in the relation by (λ, c). Under the traditional definition of subsequential transducers, making a transducer onward without changing the function it recognizes sometimes requires changing its structure by adding a new initial state (cf. Figure 1). This can happen when there are incoming transitions to the initial state. It it precisely this situation we avoid using DSFSTs, which by definition, have no incoming transitions to the initial state (condition 3, Definition 2).

Proof [Theorem 4] Since τ is a DSFST, q_0 has only one transition to one other state. So we can always push back to this transition without creating non-onwardness. We can cycle through the other states pushing back the `lcp`. The number of times we have to push the `lcp` is finite. This is because the number of times we might need to push from a state q is limited by the length of the output function of state q. Therefore, pushing will lead to an onward DSFST without, by Lemma 6, changing the transduction. ■

The DSFST of Figure 3 corresponds to the SFST of Figure 2 after applying `push_lcp` once to state 1, and the DSFST of Figure 4 shows the onward result of recursively applying `push_lcp` according to the proof of Theorem 4.

In the rest of the paper, we are only considering onward, trimmed, delimited transducers.

5. Learning Paradigm

The capability of the algorithm detailed in Section 8 is demonstrated in the identification in the limit learning paradigm [Gold, 1967], with polynomial bounds on time and data [de la Higuera, 1997]. The underlying idea of the paradigm is that if the data available to the algorithm so far does not contain enough information to distinguish the target from other potential targets, then it is impossible to learn.

We first need to define the following notions. A class $\mathbb{T}$ of functions is represented by a class $\mathbb{R}$ of representations if every $r \in \mathbb{R}$ is of finite size and there is a naming function $\mathcal{L} : \mathbb{R} \to \mathbb{T}$ which is both total and surjective. We observe that the class of subsequential functions can be represented by the class of DSFSTs.

Definition 7 *Let $\mathbb{T}$ be a class of functions represented by some class $\mathbb{R}$ of representations.*

1. *A* sample *S for a function $t \in \mathbb{T}$ is a finite set of data consistent with t, that is to say $(w, v) \in S$ iff $t(w) = v$. The size of a sample S is the sum of the length of the strings it is composed of: $|S| = \sum_{(w,v)\in S} |w| + |v|$.*
2. *A* $(\mathbb{T}, \mathbb{R})$-learning algorithm *$\mathfrak{A}$ is a program that takes as input a sample for a function $t \in \mathbb{T}$ and outputs a representation from $\mathbb{R}$.*

The paradigm relies on the notion of characteristic sample, adapted here for functions:

Definition 8 (Characteristic sample) *For a $(\mathbb{T}, \mathbb{R})$-learning algorithm $\mathfrak{A}$, a sample CS is a* characteristic sample *of a function $t \in \mathbb{T}$ if for all samples S for t such that $CS \subseteq S$, $\mathfrak{A}$ returns a representation r such that, for all $w \in \mathtt{dom}(t)$, $r(w) = t(w)$.*

This definition is the one used in the proof of the OSTIA algorithm. We are actually using the strongest version:

Definition 9 (Strong characteristic sample) *For a $(\mathbb{T}, \mathbb{R})$-learning algorithm $\mathfrak{A}$, a sample CS is a* strong characteristic sample *of a representation $r \in \mathbb{R}$ if for all samples S for $\mathcal{L}(r)$ such that $CS \subseteq S$, $\mathfrak{A}$ returns r.*

The learning paradigm can now be defined as follows.

Definition 10 (Strong identification in polynomial time and data) *A class $\mathbb{T}$ of functions is strongly identifiable in polynomial time and data (SIPTD) if there exists a $(\mathbb{T}, \mathbb{R})$-learning algorithm $\mathfrak{A}$ and two polynomials $p()$ and $q()$ such that:*

1. *For any sample S of size m for $t \in \mathbb{R}$, $\mathfrak{A}$ returns a hypothesis $r \in \mathbb{R}$ in $\mathcal{O}(p(m))$ time.*
2. *For each representation $r \in \mathbb{R}$ of size k, there exists a strong characteristic sample of r for $\mathfrak{A}$ of size at most $\mathcal{O}(q(k))$.*

For a discussion about (weak and) strong identification in the limit, the reader is referred to Clark [2014].

6. Target Classes

Output-empty transducers provide the *a priori* structural information that will allow learning of the function. A DSFST is *output-empty* if all of its transition outputs are blanks ($\square$).

Definition 11 *An* output-empty transducer $\tau_\square$ *is a DSFST* $\langle Q, q_0, q_f, \Sigma, \{\square\}, \delta\rangle$ *such that for all* $(q, a, u, q') \in \delta$, $u = \square$.

An output-empty transducer $\tau_\square$ defines a *class* of functions $\mathcal{T}$ which is exactly the set of functions which can be created by taking the states and transitions of $\tau_\square$ and replacing the blanks with output strings, maintaining onwardness.

Definition 12 *The class of functions* $\mathcal{T}$ *described by an output-empty DSFST* $\tau_\square = \langle Q, q_0, q_f, \Sigma, \{\square\}, \delta\rangle$ *is:*

$$\mathcal{T}_{\tau_\square} = \Big\{ t \mid \exists \tau = \langle Q, q_0, q_f, \Sigma, \Delta, \delta_\tau\rangle \quad \text{s.t. } \forall (q,w,v,r) \in \delta_\tau, (q,w,\square,r) \in \delta \text{ and } \forall (q,w,\square,r) \in \delta, \exists v \in \Delta^* : (q,w,v,r) \in \delta_\tau; \; t(\tau) \text{ and } onward(\tau) \Big\}$$

Lemma 13 *For all* $\tau_\square$, *all* $t \in \mathcal{T}_{\tau_\square}$ *are subsequential functions.*

Proof From Definitions 1, 11, and 12. ■

We observe there are infinitely many DSFSTs and therefore infinitely many such classes. Also, for each $\tau_\square$, each class $\mathcal{T}_{\tau_\square}$ contains infinitely many functions since there is no upper bound on the length of output strings.

7. The Inference Algorithm

The Structured Onward Subsequential Function Inference Algorithm (SOSFIA) takes as input an output-empty transducer $\tau_\square$, and a finite sample S from a target function $t \in \mathcal{T}_{\tau_\square}$. SOSFIA then iterates through the states of $\tau_\square$. At each state, it sets the output of each outgoing transition to be the *minimal change in the output* generated by this transition, according to S. To calculate this, we first define the *common output* of an input prefix w.

Definition 14 *The* common output *of an input prefix* w *in a sample* $S \subset \Sigma^* \times \Delta^*$ *for* t *is the* `lcp` *of all* $t(wv)$ *that are in* S: $\texttt{common\_out}_S(w) = \texttt{lcp}(\{u \in \Sigma^* \mid \exists v \text{ s.t. } (wv, u) \in S\})$

The minimal change in the output is then simply the difference between the common outputs of w and $w\sigma$.

Definition 15 *The* minimal change in the output *in* $S \subset \Sigma^* \times \Delta^*$ *from* w *to* $w\sigma$ *is:*

$$\texttt{min\_change}_S(\sigma, w) = \begin{cases} \texttt{common\_out}_S(\sigma) & \text{if } w = \lambda \\ \texttt{common\_out}_S(w)^{-1}\texttt{common\_out}_S(w\sigma) & \text{otherwise} \end{cases}$$

This gives us exactly the output needed to maintain onwardness, which will in turn guarantee that the SOSFIA converges to the correct function, provided that the sample contains enough information. Note that the minimal change is calculable for S because it is finite. The algorithm is presented below.

Data: A sample $S \subset \rtimes\Sigma^* \ltimes \times\Delta^*$, an output-empty DSFST $\tau_\square = \langle Q, q_0, q_f, \Sigma, \{\square\}, \delta\rangle$
Result: $\tau_\square$ as a DSFST with filled transitions
$F \leftarrow$ empty_Queue
Push$(F, (q_0, \lambda))$
mark(q_0)
while *F is not empty* **do**
  $(q, w) \leftarrow Shift\_First(F)$
  for $\sigma \in \Sigma \cup \{\rtimes, \ltimes\}$ *in lexicographic order* **do**
    for $\delta_i = (q, \sigma, \square, q') \in \delta$ **do**
      if *there exists* $\sigma' \neq \sigma$ *such that* $(q, \sigma', u, q'') \in \delta$ **then**
        Change δ_i to (q, σ, v, r), where $v = \texttt{min\_change}_S(w, \sigma)$
      else
        Change δ_i to (q, σ, λ, r)
      if q' *is not marked* **then**
        Push$(F, (q', w\sigma))$
        mark(q')
return $\tau_\square$;

Algorithm 1: Structured Onward Subsequential Function Inference Algorithm (SOSFIA)

The SOSFIA does a breadth-first parsing of the output-empty DSFST, storing in the queue F (a First-In/First-Out data type) seen but untreated states together with the smallest prefix that leads to each of these states: this is ensured by the lexicographic order in which the letters are considered and by the fact that only unmarked states (i.e., states that have not been discovered so far) are pushed into F. When a state is treated, all its outgoing transitions are considered. The output of this transition is set to be the result of the `min_change` function on the shortest prefix arriving at this state and the input letter of the transition, unless it is the only transition leaving this state in which case the output is set to be λ.[2]

8. The Theoretical Learning Result

Let $\tau = \langle Q, q_0, q_f, \Sigma, \delta\rangle$ be a DSFST. We define $\texttt{min\_pref}(q) = min\{w \in \rtimes\Sigma^*\ltimes : (q_0, w, u, q) \in \delta^*\}$.

Lemma 16 (Characteristic Sample) *Let $\tau_* = \langle Q, q_0, q_f, \Sigma, \delta_*\rangle$ be the target DSFST. Consider any finite sample S which meets the following requirements:*

- *For all $(q, \sigma, u, q') \in \delta_*$, $q \neq q_0$, there exists $(v\sigma v', w)$ in S, with $v = \texttt{min\_pref}(q)$,*

2. It may be possible to replace the entire if/then/else statement with only the part between *then* and *else* (so changing δ_i to (q, σ, v, r)). However, this appears to significantly complicate the identification proof, and obfuscate the central ideas and insights. Hence we present the algorithm as shown.

- *For all $q \in Q$ for which at least two outgoing transitions exist, for all $(q, \sigma, s, q') \in \delta_*$, if there exists $q_1, \ldots, q_n \in Q$ such that*
  1. *$q' = q_1$, and*
  2. *$\forall i < n$, q_i has only one outgoing transition and $(q_i, \sigma_i, \lambda, q_{i+1}) \in \delta_*$, and*
  3. *$(q_{n-1}, \sigma_n, s, p) \in \delta$, and*
  4. *there exist $\sigma' \neq \sigma_n$ such that $(q_{n-1}, \sigma', s', p') \in \delta_*$ and $\texttt{lcp}(s, s') = \lambda$*

  then there exists $(v\sigma\sigma_1 \ldots \sigma_n w, z)$ and $(v\sigma\sigma_1 \ldots \sigma_{n-1}\sigma' w', z')$ in S, with both being an element of the target function and $v = \texttt{min\_pref}(q)$.

Any sample S fulfilling these requirements is a strong characteristic sample for the SOSFIA.

Notice that the third and fourth items are possible only for onward transducers.

Proof Suppose that a sample CS fulfilling the requirements of Lemma 16 is contained in S. Let $\tau_A = \langle Q, q_0, q_f, \Sigma, \delta_A \rangle$ be the transducer returned by the algorithm on S. To establish strong identification, it is sufficient to show that the output of every transition in δ_A is correct (since other parts of τ_A are given by $\tau_\square$). Let $(q, \sigma, s, q') \in \delta_*$ and $v = \texttt{min\_pref}(q)$.

If there is no other outgoing transition from q, then $s = \lambda$ as τ_A is onward. By the definition of the algorithm, $(q, \sigma, \lambda, q') \in \delta_A$.

Suppose now that there exist several transitions from the state q. By construction of the sample (first item), there exists $(v\sigma v', u)$ in S. Therefore, by the definition of the learning algorithm: $(q, \sigma, \texttt{min\_change}_S(v, \sigma), q') \in \delta_A$.

Suppose $q \neq q_0$. As q admits several outgoing transitions, there exists (q, σ_1, s_1, q_1) and (q, σ_2, s_2, q_2) in δ_* such that $\texttt{lcp}(s_1, s_2) = \lambda$. By construction of the sample, $\exists v_1, v_2, u_1, u_2$ such that $(v\sigma_1 v_1, u_1)$ and $(v\sigma_2 v_2, u_2)$ are in the sample. We then have $\texttt{lcp}(u_1, u_2) = x$ and therefore $\texttt{common\_out}_S(v) = x$.

By the definition of a function, $xs \in \texttt{pref}(\texttt{common\_out}_S(v\sigma))$. If $q' = q_f$ then $(v\sigma, xs)$ is in S as there is no outgoing transition from the final state. Therefore $xs = \texttt{common\_out}_S(v\sigma))$ and then $\texttt{min\_change}(v, \sigma) = s$.

Suppose now that $q \neq q_f$. By construction, there exists $(v\sigma\sigma_1 \ldots \sigma_n w, u)$ and $(v\sigma_1 \ldots \sigma_{n-1}\sigma' w', u')$ in the sample. We have $(q', \sigma_1 \ldots \sigma_{n-1}, \lambda, q'') \in \delta_*^*$, therefore $u = xss_1v_1$ and $u' = xss_2v_2$, where $(q'', \sigma_n, s_1, p) \in \delta_*$ and $(q'', \sigma', s_2, p') \in \delta_*$ and $\texttt{lcp}(s_1, s_2) = \lambda$. Therefore $xs = \texttt{common\_out}_S(v\sigma)$ and then $\texttt{min\_change}(v, \sigma) = s$. Hence, $(q, \sigma, s, q') \in \delta_A$.

The proof for the case $q = q_0$ is of similar nature. ■

Lemma 17 (Time Complexity) *Consider a target transducer τ and a characteristic sample S and let m be the length of the longest string in the right projection of S, and $n = |S|$. The overall time complexity for the SOSFIA is $O(n \cdot m)$.*

Proof The *while* loop is executed exactly $|Q|$ times since every state is considered once and only once. Then the two *for* loops are executed overall $|\Sigma| \cdot card(\delta)$ times since each transition is modified only once ($card(\delta)$ is the cardinality of δ). In the worst case, the algorithm launches `min_change` for each transition, which corresponds to the computation of two `lcp`. Each of these calculations is doable in $O(n \cdot m)$ with an adapted structure for storing strings (e.g., prefix tree acceptor). The overall time complexity of the algorithm is

thus in $O(n \cdot m)$ since $|Q|$ and $card(\delta)$ are fixed. ■

Lemma 18 (Data Complexity) *The size of a characteristic sample defined in Lemma 16 for a target DSFST $\tau = \langle Q, q_0, q_f, \Sigma, \Delta, \delta \rangle$ is $O(|\tau|)$.*

Proof One needs at most $|Q|$ elements to fulfill the first requirement. Each of these string pairs can have a left projection of length at most $|Q|$ and a right projection of length at most $\sum_{(q,\sigma,u,q') \in \delta} |u|$. This is due to the fact that in the worst case these elements correspond to a parsing through all states and all transitions of τ (in case τ admits cycles, longer elements can be considered, but we focus only on the smallest ones in this proof). Therefore the size of this part of the characteristic sample is in $O(|Q| \cdot (|Q| + |\tau|)) = O(|\tau|)$ since $|Q|$ is a constant for each target class.

The number of elements needed for a sample to fulfill the second item of the definition is at most $2 \cdot card(\delta)$. For the same reason as previously, these elements can have a left projection of length at most $|Q|$ and a right projection of length at most $\sum_{(q,\sigma,u,q') \in \delta} |u|$. The size of this second part of the characteristic sample is thus in $O(card(\delta) \cdot |\tau|) = O(|\tau|)$ as the cardinality of δ is fixed for each target class. ■

Theorem 19 *For every output-empty transducer $\tau_\square$, the SOSFIA strongly identifies $\mathcal{T}_{\tau_\square}$ in polynomial time and data.*

Proof For any $\tau_\square$ and any $\tau \in \mathcal{T}_{\tau_\square}$, Lemma 16 establishes that SOSFIA outputs τ for any sample which includes a characteristic sample CS of τ. Lemma 17 establishes that SOSFIA produces this output in linear time with respect to the size of CS. Lemma 18 establishes the size of CS is linear in the size of τ. ■

It is worth mentioning that the algorithm presented in this paper returns a function even given incomplete data. With a sacrifice in added time complexity linear to the sum of the length of the left projection of the sample, the algorithm can be modified so that it checks if any element of the sample uses a transition it is about to modify. If this is not the case it can stop since it is sure that the data is not sufficient for learning. We have opted not to include this functionality, as this is not enough to guarantee that there is sufficient data to calculate the output for the transition (as it is still unknown whether *all* the input/output pairs necessary to calculate the correct output for that transition are in the data). Further work may explore efficient ways of giving the learner this functionality.

It should be interesting to also compare the subclasses of subsequential transductions learnable with SOSFIA and those with OSTIA-D, OSTIA-R, and OSTIA-DR. We conjecture that SOSFIA and OSTIA-D correspond to the same class of learnable transductions.

9. Demonstrations

The applicability of subsequential transducers to natural language phonology and morphology has been discussed in detail bt Mohri [1997] and Beesley and Karttunen [2003]. Briefly,

morphological operations can be thought of as a transformation from a bare form to an affixed form, and a foundational principle of modern generative phonology is that there is a phonological mapping from abstract, lexical 'underlying' representations of words and morphemes to their concrete surface pronunciations [Hayes, 2011]. In this section we present several demonstrations of the SOSFIA using linguistically-motivated examples.

9.1. Input Strictly Local Functions

As mentioned above, Chandlee [2014] shows that the input-output mapping of many phonological processes can be modeled with Input Strictly Local (ISL) functions. The automata-theoretic characterization of these functions provides sufficient structure for the SOSFIA to identify them in the limit. We tested the learner on three example ISL functions, all of which can be thought of as phonological 'repairs' that prevent a surface output sequence D$\ltimes$ (where D is a voiced obstruent). One is to 'devoice' a final voiced sound (e.g., a 'd' sound), changing instead to a voiceless one (e.g., a 't' sound). The others are to delete the voiced sound or to epenthesize (insert) a vowel in between the voiced sound and the word edge. The examples are summarized in (1).

(1)

Process	Rule
Final devoicing	D $\rightarrow$ T / $\_\_\ltimes$
Deletion	D $\rightarrow \emptyset$ / $\_\_ \ltimes$
Epenthesis	$\emptyset \rightarrow$ V / D$__ \ltimes$

Using an alphabet that represents common sound categories $\Sigma = \{$D, T, N, V$\}$, where again D is a voiced obstruent (e.g., 'b', 'd', 'z', 'g') T is a voiceless obstruent (e.g., 'p', 't', 's', 'k'), N is a sonorant consonant (nasal sounds and 'l' and 'r'), and V is a vowel, we constructed a data set of 1365 string pairs. The left projection of the data set is $\Sigma^{\leq 5}$; each string in the left projection was paired with an output string according to the target function. The same output-empty transducer was used in all three test cases, as the DSFST for each rule only differs in terms of the output strings. The output of the SOSFIA is correct in all three cases; as an example, the resulting machine for the final devoicing test case is given in Figure 5. The machines for the deletion and epenthesis cases were identical, except for that in the deletion case the transition from state 2 to state 6 has an output of λ, and in the epenthesis case every transition into state 2 on D has an output of D and the transitions out of state 2 are as follows: $(2, D, D, 2), (2, T, T, 3), (2, V, V, 5), (2, N, N, 4), (2, \ltimes, V, 6)$.

9.2. Non-ISL Phonological Processes

There also exist phonological processes that cannot be modeled with ISL functions. These are 'long-distance' processes such as the sibilant harmony process attested in Samala. In this language, all sibiliant sounds (e.g., 's', 'ʃ') in a word must be the same as the rightmost one [Applegate, 1972; Hansson, 2010]. An example is shown in (2).

(2) /hasxintilawaʃ/ $\mapsto$ [haʃxintilawaʃ] 'his former gentile name'

Using the (simplified) alphabet $\Sigma = \{$s, ʃ, t, a$\}$, we constructed a dataset of string pairs in which the left projection is $\Sigma^{\leq 4}$. Each of these strings was paired with one in which all

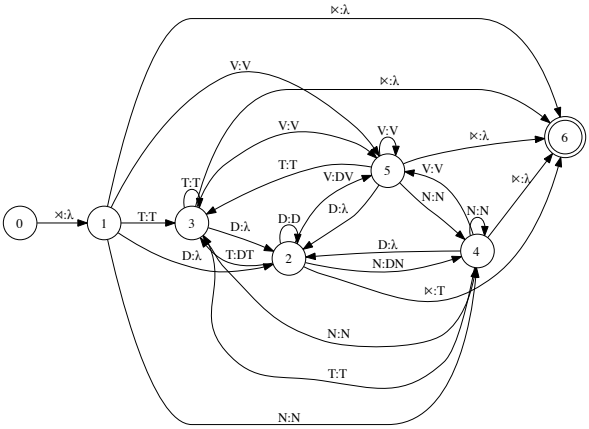

Figure 5: Final devoicing test case

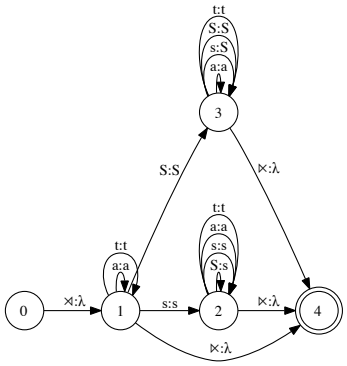

Figure 6: Sibilant harmony test case

sibiliants (if any) assimilate to the rightmost one. As shown in Figure 6, the output of the SOSFIA correctly models this process.

Two things should be noted about this test case. One is that this process is right subsequential; however, all this entails is that the input and output strings must be reversed to get the correct mapping. This has no effect on how the learner functions (for details on right subsequential functions in phonology, see [Heinz and Lai, 2013]). Two, the output-empty DSFST given to the learner includes separate states for words in which 's' is the rightmost sibiliant and words in which 'ʃ' is the rightmost sibilant (i.e., the *a priori* knowledge given to the learner identifies the set of segments involved in the process). This is akin to phonological theories in which certain classes of segments (here the sibilants) are represented on distinct tiers (e.g., vowel harmony; [Clements, 1976]).

9.3. Morphological Parsing

The learner was also tested on a morphology-to-phonology function, as discussed in [Beesley and Karttunen, 2003], in which the meanings of morphemes are mapped to their pronunciations. The data were from Swahili verbs [Hayes, 2011], which show a series of prefixes indicating person, number, and tense. The morphological breakdown of *nimenipenda* 'I have liked myself' is given in (3).[3] The learner was given an empty transducer representing the possible morpheme orders and 90 pairs of the shape <*morpheme string, phoneme string*>, as exemplified in (4). The learner correctly learned the morphological function; i.e., for any transition on '1-nom' it learned that the output should be 'ni'.

(3) ni + me + ni + penda
1-acc perf 1st-nom like
'I have liked myself'

(4) 1-nom+perf+1-acc+like, nimenipenda
3-nom+pres+1-acc+like, ananipenda
2-nom+perf+1-pl-acc+beat, umetupiga

3. Abbreviations: 1st-acc = first person accusative ('me'); 1-nom = first person nominative ('I'); 1-pl-acc = first person plural accusative ('us'); 2-nom = second person nominative ('we'); 3-nom = third person nominative ('he'); pres = present tense; perf = perfect tense.

10. Conclusion

This paper has made two contributions. One, we introduced DSFSTs and showed that, unlike SFSTs, they can be made onward without modifying their structure. Second, we introduced a learning algorithm for proper subclasses of subsequential functions in which all member functions share a DSFST structure. There are infinitely many such classes, each with infinitely many member functions. While the learner is given a significant amount of prior knowledge in the form of the output-empty transducer representing the class of functions, this allows for extremely efficient learning measured in both time and data. We have also shown specific applications for the learner in the linguistic domains of phonology and morphology.

Acknowledgments

We thank reviewers of this paper for their helpful comments. This research was supported by NSF grant CPS#1035577.

References

Richard Applegate. *Inseño Chumash grammar*. PhD thesis, University of California, Berkeley, 1972.

Kenneth R. Beesley and Lauri Karttunen. *Finite State Morphology*. CSLI Publications, 2003.

Antonio Castellanos, Enrique Vidal, Miguel A. Varó, and José Oncina. Language understanding and subsequential transducer learning. *Computer Speech and Language*, 12: 193–228, 1998.

Jane Chandlee. *Strictly Local Phonological Processes*. PhD thesis, University of Delaware, 2014.

Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz. Learning strictly local subsequential functions. *Transactions of the Association for Computational Linguistics*, to appear.

Alexander Clark. Inference of inversion transduction grammars. In *ICML*, pages 201–208, 2011.

Alexander Clark. Learning trees from strings: A strong learning algorithm for some context-free grammars. *Journal of Machine Learning Research*, 14:3537–3559, 2014.

Georges N. Clements. *Vowel harmony in nonlinear generative phonology: an autosegmental model*. Indiana University Linguistics Club, 1976.

Colin de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning Journal*, 27:125–138, 1997.

Colin de la Higuera. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, 2010.

Pedro García, Enrique Vidal, and José Oncina. Learning locally testable languages in the strict sense. In *Proceedings of the Workshop on Algorithmic Learning Theory*, pages 325–338, 1990.

Mark E. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

Gunnar Ólafur Hansson. *Consonant harmony: long-distance interaction in phonology*. Berkeley, CA: University of California Press, 2010.

Bruce Hayes. *Introductory Phonology*. John Wiley & Sons, 2011.

Jeffrey Heinz and Regine Lai. Vowel harmony and subsequentiality. In Andras Kornai and Marco Kuhlmann, editors, *Proceedings of the 13th Meeting on Mathematics of Language*, Sofia, Bulgaria, 2013.

Jeffrey Heinz and James Rogers. Learning subregular classes of languages with factored deterministic automata. In Andras Kornai and Marco Kuhlmann, editors, *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, pages 64–71, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

Robert McNaughton and Seymour Papert. *Counter-Free Automata*. MIT Press, 1971.

Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

José Oncina and Pedro García. Inductive learning of subsequential functions. Technical Report DSIC II-34, Univ. Politécnia de Valencia, 1991.

José Oncina and Pedro García. *Inferring regular languages in polynomial update time*, chapter -. World Scientific Publishing, 1992.

José Oncina and Miguel A. Varò. Using domain information during the learning of a subsequential transducer. *Lecture Notes in Computer Science - Lecture Notes in Artificial Intelligence*, pages 313–325, 1996.

José Oncina, Pedro García, and Enrique Vidal. Learning subsequential transducers for pattern recognition tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:448–458, May 1993.